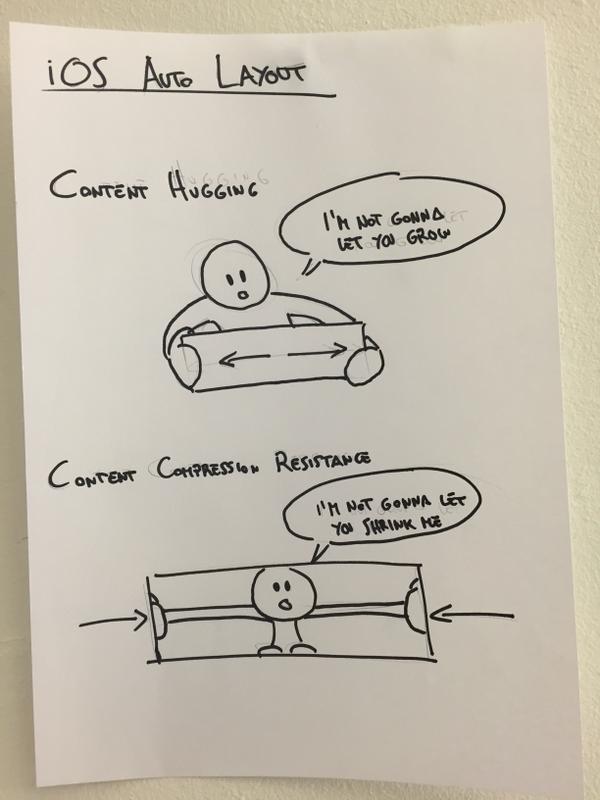
# Basic Exercises Part 3.1 UILabels and Priorities.

* **Content Hugging** Priority - The higher this priority **is**, the more a view resists growing larger than its intrinsic **content** size. **Content Compression Resistance** Priority - The higher this priority **is**, the more a view resists shrinking smaller than its intrinsic **content** size.



A quick summary of the concepts:

* Hugging => content does not want to grow
* Compression Resistance => content does not want to shrink

Example:

Say you've got a button like this:

[ Click Me ]

and you've pinned the edges to a larger superview with priority 500.

Then, if Hugging priority > 500 it'll look like this:

[Click Me]

If Hugging priority < 500 it'll look like this:

[ Click Me ]

If the superview now shrinks then, if the Compression Resistance priority > 500, it'll look like this

[Click Me]

Else if Compression Resistance priority < 500, it could look like this:

[Cli..]

If it doesn't work like this then you've probably got some other constraints going on that are messing up your good work!

E.g. you could have it pinned to the superview with priority 1000. Or you could have a width priority. If so, this can be helpful:

Editor > Size to Fit Content

## UILABEL

### **1.1 Programmatically!!**

Write a UIViewController subclass in Swift that contains a UILabel that displays the current date.

### **1.2 Now ObjC**

Write an Objective-C view controller that has the same behavior as your Swift view controller.

Hints:

**#1** don’t bother about the constraints.

## #2

## Swift

let frame = CGRect(x: 0, y: 0, width: 200, height: 21)

let label = UILabel(frame: frame)

view.addSubview(label)

## Objective-C

CGRect frame = CGRectMake(0, 0, 200, 21);

UILabel \*label = [[UILabel alloc] initWithFrame:frame];

[view addSubview:label];

#3. Try to positioning your label in the center. Set some more properties. Background color to clear color. Text color red. Fon Verdana size 18. Number of lines 2. Shadow color to dark gray.

#4. Note the fact to add to the subviews you new element in the UI hierarchy.

#5. Apply the styles using a function.

#6. What is the stands for: alloc-initWithFrame. (**If you want to create an instance of a specific subclass, you must alloc/init the UI element directly**).

### **1.3 In a row.**

Create multiple labels in a single row.

### **1.4 Initialize them**

Initialize your labels in the next lines.

### **1.5 Using a loop**

Create and Initialize 5 labels with a loop (your choice). Remember to add some properties and presenting over the simulator.

### **1.6 Stack them**

Put your labels inside a stack view (horizontal and vertical). This could be done in a new project directly in the builder interface.

### **1.7 Click**

Make a UILabel tapable. This is not a good idea but try out. Any reason to detect touches on words in UILabels?

1. Create label
2. Enable user interaction
3. Add UITapGestureRecognizer

The key to create a clickable UILabel is to enable user interaction. Don’t forget to create the function and simply print something in the console.

### **1.8 UILabel and intrinsic content size.**

All views have an intrinsic content size, which refers to the amount of space the view needs for its content to appear in an ideal state. For example, the intrinsic content size of a **UILabel** will be the size of the text it contains using whatever font you have configured it to use.

Intrinsic content sizes are important because they allow views to have a natural width and height without us forcing one. For Auto Layout to work it must know where each view is positioned precisely: its X, Y, width, and height values. With intrinsic content size we can say “place this button 20 points from the top and center it horizontally” and that’s enough to form a complete layout – Auto Layout can calculate the rest based on the button’s intrinsic size.

Although Auto Layout does its best to give views the space they need based on their intrinsic content sizes, all views also have a content compression resistance priority and a content hugging priority that determine how much it fights to retain its intrinsic content size when available space is less than or greater than it needs, respectively.

### **1.9 Apple’s documentation.**

<https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/ViewswithIntrinsicContentSize.html>

### **1.10 Tutorial**

Complete this tutorial.

<https://fluffy.es/what-is-intrinsic-content-size/>